Attorney Docket No.: P00-2941
Client/Matter No.: 68854.0133
Express Mail No.: EL384120842US

METHOD FOR TRANSACTION COMMAND ORDERING IN A REMOTE DATA REPLICATION SYSTEM

FIELD OF THE INVENTION

The present invention relates generally to data consistency in data storage systems, and more specifically, to a method for pipelining a number of write commands between a sending site and a receiving site while providing command ordering during controller-based synchronous or asynchronous copy operations in a remote data replication system using a Storage Area Network.

BACKGROUND OF THE INVENTION AND PROBLEM

It is desirable to provide the ability for rapid recovery of user data from a disaster or significant error event at a data processing facility. This type of capability is often termed 'disaster tolerance'. In a data storage environment, disaster tolerance requirements include providing for replicated data and redundant storage to support recovery after the event. In order to provide a safe physical distance between the original data and the data to backed up, the data must be migrated from one storage subsystem or physical site to another subsystem or site. It is also desirable for user applications to continue to run while data replication proceeds in the background. Data warehousing, 'continuous computing', and Enterprise Applications all require remote copy capabilities.

Storage controllers are commonly utilized in computer systems to off-load from the host computer certain lower level processing functions relating to I/O operations, and to serve as interface between the host computer and the physical storage media. Given the critical role played by the storage controller with respect to computer system I/O performance, it is desirable to minimize the potential for interrupted I/O service due to storage controller malfunction. Thus, prior workers in the art have developed various system design approaches in an attempt to achieve some degree of fault tolerance in the storage control function. One such prior approach requires that all system functions be "mirrored". While this type of approach is most effective in reducing interruption of I/O operations and lends itself to value-added fault isolation techniques, it has previously been costly to implement and heretofore has placed a heavy processing burden on the host computer.

One prior method of providing storage system fault tolerance accomplishes failover through the use of two controllers coupled in an active/passive configuration. During failover, the passive controller takes over for the active (failing) controller. A drawback to this type of dual configuration is that it cannot support load balancing, as only one controller is active and thus utilized at any given time, to increase overall system performance. Furthermore, the passive controller presents an inefficient use of system resources.

Another approach to storage controller fault tolerance is based on a process called 'failover'. Failover is known in the art as a process by which a first storage controller, coupled to a second controller, assumes the responsibilities of the second controller when the second controller fails. 'Failback' is the reverse operation, wherein the second controller, having been either repaired or replaced, recovers control over its originally-attached storage devices. Since each controller is capable of accessing the storage devices attached to the other controller as a result of the failover, there is no need to store and maintain a duplicate copy of the data, i.e., one set stored on the first controller's attached devices and a second (redundant) copy on the second controller's devices.

U.S. Patent No. 5,274,645 (Dec. 28, 1993), to Idleman et al. discloses a dual-active configuration of storage controllers capable of performing failover without the direct involvement of the host. However, the direction taken by Idleman requires a multi-level storage controller implementation. Each controller in the dual-redundant pair includes a two-level hierarchy of controllers. When the first level or host-interface controller of the first controller detects the failure of the second level or device interface controller of the second controller, it re-configures the data path such that the data is directed to the functioning second level controller of the second controller. In conjunction, a switching circuit re-configures the controller-device interconnections, thereby permitting the host to access the storage devices originally connected to the failed second level controller through the operating second level controller of the second controller. Thus, the presence of the first level controllers serves to isolate the host computer from the failover operation, but this isolation is obtained at added controller cost and complexity.

Other known failover techniques are based on proprietary buses. These techniques utilize existing host interconnect "hand-shaking" protocols, whereby the host and controller act in cooperative effort to effect a failover operation. Unfortunately, the "hooks" for this and other types of host-assisted failover mechanisms are not compatible with more recently developed, industry-standard interconnection protocols, such as SCSI, which were not developed with failover capability in mind. Consequently, support for dual-active failover in these proprietary bus techniques must be built into the host firmware via the host device drivers. Because SCSI, for example, is a popular industry standard interconnect, and there is a commercial need to support platforms not using proprietary buses, compatibility with industry standards such as SCSI is essential. Therefore, a vendor-unique device driver in the host is not a desirable option.

However, none of the above references disclose a disaster tolerant data storage system having a remote backup site connected to a host site via a dual fabric link, where the system replication and error recovery functions are controller-based. Furthermore, none of the above systems allows a number of write commands to be 'pipelined' (in transit and unacknowledged) between local and remote sites while ensuring the proper ordering of commands on remote media during synchronous or asynchronous operation. In addition, the prior technology fails to provide a mechanism for 'tuning' of links based on distance and performance requirements.

Therefore, there is a clearly felt need in the art for a disaster tolerant data replication system capable of optimally tunable inter-site performance, and which allows commands to be pipelined during operation, where the data replication functions are performed without the direct involvement of the host computer.

SOLUTION TO THE PROBLEM

Accordingly, the above problems are solved, and an advance in the field is accomplished by the system of the present invention which provides a completely redundant configuration including dual Fibre Channel fabric links interconnecting each of the components of two data storage sites, wherein each site comprises a host computer and associated data storage array, with redundant array controllers and adapters. The present system is unique in that each array controller is capable of performing all of the data replication functions including the handling of failover functions.

In the situation wherein an array controller fails during an asynchronous copy operation, the partner array controller uses a 'micro log' stored in mirrored cache memory to recover transactions which were 'missed' by the backup storage array when the array controller failure occurred. The present system provides rapid and accurate recovery of backup data at the remote site by sending all logged commands and data from the logging site over the link to the backup site in order, while avoiding the overhead of a full copy operation.

An important aspect of the present invention is the concept of a 'look-ahead limit', which, as implemented, allows a large number of 'outstanding' commands to be concurrently be in transit between sites at any given time, while guaranteeing in-order operation of the data replication function. In addition, the present system automatically calculates an average transit/response time for each specific link, which is employed to determine a worst-case response time for effecting failover operations. A parameter representing the number of outstanding commands is user-adjustable, which allows tuning of links based on distance and system performance requirements.

The 'mirroring' of data for backup purposes is the basis for RAID ('Redundant Array of Independent [or Inexpensive] Disks') Level 1 systems, wherein all data is replicated on N separate disks, with N usually having a value of 2. Although the concept of storing copies of data at a long distance from each other (i.e., long distance mirroring) is known, the use of a switched, dual-fabric, Fibre Channel configuration as described herein is a novel approach to disaster tolerant storage systems. Mirroring requires that the data be consistent across all volumes. In prior art systems which use host-based mirroring (where each host computer sees multiple units), the host maintains consistency across the units. For those systems which employ controller-based mirroring (where the host computer sees only a single unit), the host is not signaled completion of a command until the controller has updated all pertinent volumes. The present invention is, in one aspect, distinguished over the previous two types of systems in that the host computer sees multiple volumes, but the data replication function is performed by the controller. Therefore, a mechanism is required to communicate the association between volumes to the controller. To maintain this consistency between volumes, the system of the present invention provides a mechanism of associating a set of volumes to synchronize the logging to the set of volumes so that when the log is consistent when it is "played back" to the remote site.

Each array controller in the present system has a dedicated link via a fabric to a partner on the remote side of the long-distance link between fabric elements. Each dedicated link does not appear to any host as an available link to them for data access, however, it is visible to the partner array controllers involved in data replication operations. These links are managed by each partner array controller as if being 'clustered' with a reliable data link between them.

The fabrics comprise two components, a local element and a remote element. An important aspect of the present invention is the fact that the fabrics are 'extended' by standard e-ports (extension ports). The use of e-ports allow for standard Fibre Channel cable to be run between the fabric elements or the use of a conversion box to covert the data to a form such as telco ATM or IP. The extended fabric allows the entire system to be viewable by both the hosts and storage.

The dual fabrics, as well as the dual array controllers, dual adapters in hosts, and dual links between fabrics, provide high-availability and present no single point of failure. A distinction here over the prior art is that previous systems typically use other kinds of links to provide the data replication, resulting in the storage not being readily exposed to hosts on both sides of a link. The present configuration allows for extended clustering where local and remote site hosts are actually sharing data across the link from one or more storage subsystems with dual array controllers within each subsystem.

The present system is further distinguished over the prior art by other additional features, including independent discovery of initiator to target system and automatic rediscovery after link failure. In addition, device failures, such as controller and link failures, are detected by 'heartbeat' monitoring by each array controller. Furthermore, no special host software is required to implement the above features because all replication functionality is totally self contained within each array controller and automatically done without user intervention.

An additional aspect of the present system is the ability to function over two links with data replication traffic. If failure of a link occurs, as detected by the 'initiator' array controller, that array controller will automatically 'failover', or move the base of data replication operations to its partner controller. At this time, all transfers in flight are discarded, and therefore discarded to the host. The host simply sees a controller failover at the host OS (operating system) level, causing the OS to retry the operations to the partner controller. The array controller partner continues all 'initiator' operations from that point forward. The array controller whose link failed will continuously watch that status of its link to the same controller on the other 'far' side of the link. That status changes to a 'good' link when the array controllers have established reliable communications between each other. When this occurs, the array controller 'initiator' partner will 'failback' the link, moving operations back to newly reliable link. This procedure re-establishes load balance for data replication operations automatically, without requiring additional features in the array controller or host beyond what is minimally required to allow controller failover.

BRIEF DESCRIPTION OF THE DRAWINGS

The above objects, features and advantages of the present invention will become more apparent from the following detailed description taken in conjunction with the accompanying drawings, in which:

FIG. 1 is a diagram showing long distance mirroring;

FIG. 2 illustrates a switched dual fabric, disaster-tolerant storage system;

FIG. 3 is a block diagram of the system shown in FIG. 2;

FIG. 4 is a high-level diagram of a remote copy set operation;

FIG. 5 is a block diagram showing exemplary controller software architecture;

FIG. 6A is a flow diagram showing inter-site controller heartbeat timer operation;

FIG. 6B is a flow diagram showing intra-site controller heartbeat timer operation;

FIG. 7 is a flowchart showing synchronous system operation;

FIG. 8A is a flowchart showing asynchronous system operation;

FIG. 8B is a flowchart showing a 'micro-merge' operation;

FIG. 9 is a diagram showing an example of a link failover operation;

FIG. 10 is a flow diagram showing a log operation when both links are down, or when the remote site is down;

FIG. 11 is a flowchart showing log unit writing and merging operations;

FIG. 12 is a flow diagram showing a log operation in response to a site failover;

FIG. 13 is a diagram showing an exemplary format of data and extent information stored on a log unit;

FIG. 14 is a flowchart illustrating an exemplary method used by the present system to ensure proper write ordering on the remote media; and

FIG. 15 is a flowchart of a method used by the present system to calculate a worst-case response time for effecting failover operations.

DETAILED DESCRIPTION

The system of the present invention comprises a data backup and remote copy system which provides disaster tolerance. In particular, the present system provides a peer-to-peer remote copy (backup) function which is implemented as a controller-based replication of one or more LUNs (logical units) between two remotely separated pairs of array controllers connected by redundant links. The present system further provides a data logging mechanism (a write history 'log unit') for storing commands and data for every transaction that occurs in the situation where the remote backup storage device is unavailable because both links have failed, a remote site is down, or because of a site failover. The system performs an in-order merging of the log unit data with the data on the previously unavailable backup device to quickly return both local and remote sites to the same data state after link restoration or remote site restoration. In the situation wherein an array controller fails during an asynchronous copy operation, the partner array controller uses a 'micro log' stored in mirrored cache memory to recover transactions which were 'missed' by the backup storage array when the array controller failure occurred.

The present invention dynamically determines a 'look-ahead' command limit, which establishes the number of pipelined, or 'outstanding' (not-responded-to) commands which can concurrently be in transit between sites at any given time. Another important aspect of the present invention is the implementation of a method to ensure the proper ordering of commands on remote media during asynchronous data replication, while allowing a large number of commands to be pipelined between local and remote sites. In addition, the present system determines an average transit/response time for each specific link to provide the basis for calculating a user-adjustable worst-case value for link timeouts.

Figure 1 is a diagram showing long distance mirroring, which is an underlying concept of the present invention. The present system 100 employs a switched, dual-fabric, Fibre Channel configuration to provide a disaster tolerant storage system. Fibre Channel is the general name of an integrated set of standards developed by the American National Standards Institute (ANSI) which defines protocols for information transfer. Fibre Channel supports multiple physical interface types, multiple protocols over a common physical interface, and a means for interconnecting various interface types. A 'Fibre Channel' may include transmission media such as copper coax or twisted pair copper wires in addition to (or in lieu of) optical fiber.

As shown in Figure 1, when host computer 101 writes data to its local storage array, an initiating node, or 'initiator' 111 sends a backup copy of the data to remote 'target' node 112 via a Fibre Channel switched fabric 103. A 'fabric' is a topology (explained in more detail below) which supports dynamic interconnections between nodes through ports connected to the fabric. In Figure 1, nodes 111 and 112 are connected to respective links 105 and 106 via ports 109. A node is simply a device which has at least one port to provide access external to the device. In the context of the present system 100, a node typically includes an array controller pair and associated storage array. Each port in a node is generically termed an N (or NL) port. Ports 109 (array controller ports) are thus N ports. Each port in a fabric is generically termed an F (or FL) port. In Figure 1, links 105 and 106 are connected to switched fabric 103 via F ports 107. More specifically, these F ports may be E ports (extension ports) or E port/FC-BBport pairs, as explained below.

In general, it is possible for any node connected to a fabric to communicate with any other node connected to other F ports of the fabric, using services provided by the fabric. In a fabric topology, all routing of data frames is performed by the fabric, rather than by the ports. This any-to-any connection service ('peer-to-peer' service) provided by a fabric is integral to a Fibre Channel system. It should be noted that in the context of the present system, although a second host computer 102 is shown (at the target site) in Figure 1, this computer is not necessary for operation of the system 100 as described herein.

An underlying operational concept employed by the present system 100 is the pairing of volumes (or LUNs) on a local array with those on a remote array. The combination of volumes is called a 'remote copy set'. A remote copy set thus consists of two volumes, one on the local array, and one on the remote array. For example, as shown in Figure 1, a remote copy set might consist of LUN 1 (110) on a storage array at site 101 and LUN $1'$(110 $'$) on a storage array at site 102. The array designated as the 'local' array is called the initiator, while the remote array is called the target. Various methods for synchronizing the data between the local and remote array are possible in the context of the present system. These synchronization methods range from full synchronous to fully asynchronous data transmission, as explained below. The system user's ability to choose these methods provides the user with the capability to vary system reliability with respect to potential disasters and the recovery after such a disaster. The present system allows choices to be made by the user based on factors which include likelihood of disasters and the critical nature of the user's data.

System Architecture

Figure 2 illustrates an exemplary configuration of the present invention, which comprises a switched dual fabric, disaster-tolerant storage system 100. The basic topology of the present system 100 is that of a switched-based Storage Area Network (SAN). As shown in Figure 2, data storage sites 218 and 219 each respectively comprise two hosts 101/101A and 102/102A, and two storage array controllers 201/202 and 211/212 connected to storage arrays 203 and 213, respectively. Alternatively, only a single host 101/102, or more than two hosts may be connected to system 100 at each site 218/219. Storage arrays 203 and 213 typically comprise a plurality of magnetic disk storage devices, but could also include or consist of other types of mass storage devices such as semiconductor memory.

In the configuration of Figure 2, each host at a particular site is connected to both fabric elements (i.e., switches) located at that particular site. More specifically, at site 218, host 101 is connected to switches 204 and 214 via respective paths 231A and 231B; host 101A is connected to the switches via paths 241A and 241B. Also located at site 218 are array controllers A1 (ref. no. 201 and A2 (ref. no. 202). Array controller A1 is connected to switch 204 via paths 221H and 221D; array controller A2 is connected to switch 214 via paths 222H and 222D. The path suffixes 'H' and 'D' refer to 'Host' and 'Disaster-tolerant' paths, respectively, as explained below. Site 219 has counterpart array controllers B1 (ref. no 211) and B2 (ref. no. 212), each of which is connected to switches 205 and 215. Note that array controllers B1 and B2 are connected to switches 205 and 215 via paths 251D and 252D, which are, in effect, continuations of paths 221D and 222D, respectively.

In the present system shown in Figure 2, all storage subsystems (201/202/203 and 211/212/213) and all hosts (101, 101A, 102, and 102A) are visible to each other over the SAN 103A/103B. This configuration provides for high availability with a dual fabric, dual host, and dual storage topology, where a single fabric, host, or storage can fail and the system can still continue to access other system components via the SAN. As shown in Figure 2, each fabric 103A/103B employed by the present system 100 includes two switches interconnected by a high-speed link. More specifically, fabric 103A comprises switches 204 and 205 connected by link 223A, while fabric 103B comprises switches 214 and 215 connected by link 223B.

Basic Fibre Channel technology allows the length of links 223A/223B (i.e., the distance between data storage sites) to be as great as 10KM as per the FC-PH3 specification (see Fibre Channel Standard: Fibre Channel Physical and Signaling Interface, ANSII X3T11). However, distances of 20KM and greater are possible given improved technology and FC-PH margins with basic Fibre Channel. FC-BB (Fibre Channel Backbone) technology provides the opportunity to extend Fibre Channel over leased Telco lines (also called WAN tunneling). In the case wherein FC-BB is used for links 223A and 223B, FC-BB ports are attached to the E ports to terminate the ends of links 223A and 223B.

It is also possible to interconnect each switch pair 204/205 and 214/215 via an Internet link (223A/223B). If the redundant links 223A and 223B between the data storage sites 218/219 are connected to different ISPs (Internet Service Providers) at the same site, for example, there is a high probability of having at least one link operational at any given time. This is particularly true because of the many redundant paths which are available over the Internet between ISPs. For example, switches 204 and 214 could be connected to separate ISPs, and switches 205 and 215 could also be connected to separate ISPs.

Figure 3 is an exemplary block diagram illustrating additional details of the system shown in Figure 2. The configuration of the present system 100, as shown in Figure 3, depicts only one host per site for the sake of simplicity. Each host 101/102 has two adapters 308 which support the dual fabric topology. The hosts typically run multi-pathing software that dynamically allows failover between storage paths as well as static load balancing of storage volumes (LUNs) between the paths to the controller-based storage arrays 201/202 and 211/212. The configuration of system 100 allows for applications using either of the storage arrays 203/213 to continue running given any failure of either fabric 103A/103B or either of the storage arrays.

The array controllers 201/202 and 211/212 employed by the present system 100 have two host ports 109 per array controller, for a total of four connections (ports) per pair in the dual redundant configuration of Figure 3. Each host port 109 preferably has an optical attachment to the switched fabric, for example, a Gigabit Link Module ('GLM') interface at the controller, which connects to a Gigabit Converter ('GBIC') module comprising the switch interface port 107. Switch interconnection ports 306 also preferably comprise GBIC modules. Each pair of array controllers 201/202 and 211/212 (and associated storage array) is also called a storage node (e.g., 301 and 302), and has a unique Fibre Channel Node Identifier. As shown in Figure 3, array controller pair A1/A2 comprise storage node 301, and array controller pair B1/B2 comprise storage node 302. Furthermore, each storage node and each port on the array controller has a unique Fibre Channel Port Identifier, such as a World-Wide ID (WWID). In addition, each unit connected to a given array controller also has a WWID, which is the storage node's WWID with an incrementing 'incarnation' number. This WWID is used by the host's O/S to allow the local and remote units to be viewed as the 'same' storage.

The array controllers' ports 109 are connected somewhat differently than typical dual controller/adapter/channel configurations. Normally, the controller ports' connections to dual transmission channels are cross-coupled, i.e., each controller is connected to both channels. However, in the present system configuration 100, both ports on array controller A1, for example, attach directly to a single fabric via switch 204. Likewise, both ports on array controller A2 attach directly to the alternate fabric, via switch 214. The exact same relative connections exist between array controllers B1/B2 and their respective switches 205/215 and associated fabrics. One port of each controller is the 'host' port that will serve LUN(s) to the local host 101/102. The other port of each controller is the 'remote copy' port, used for disaster tolerant backup.

Remote Copy Sets

Figure 4 is a high-level diagram of a 'remote copy set' operation. The present system 100 views volumes (or LUNs) on a local array as being paired with counterpart volumes on a remote array. A remote copy set comprises a pair of same-sized volumes, one on the local array, and one on the remote array. When a local host computer 101, for example, requests a storage array I/O operation, the local array controller, or 'initiator' 301, presents a local volume that is part of the remote copy set to the local host. The host 101 performs writes to the local volume on the local array 203 , which copies the incoming write data to the remote volume on the target array 213.

As shown in Figure 4, two LUNs (logical units), LUN X (410) and LUN X′(410′), attached to controllers B1/B2 (302) and A1/A2 (301), respectively, are bound together as a remote copy set 401. A remote copy set (RCS), when added on array 203, points to array 213, and will cause the contents of the local RCS member on array 203 to be immediately copied to the remote RCS member on array 213. When the copy is complete, LUN X′(410′) on array 213 is ready to be used as a backup device. In order to preserve the integrity of the backup copy, local host 101 access to LUN 410′ is not allowed during normal operations.

Software Architecture

Figure 5 is a block diagram showing exemplary array controller software architecture employed by the present system 100. As shown in Figure 5, peer-to-peer remote copy software ('PPRC manager') 515 is layered in between host port initiator module 510 and VA ('Value Added', such as RAID and caching) software module 520 within each controller (A1/A2/B1/B2). VA layer 520 is not aware of any PPRC manager 515 context (state change or transfer path). Host port target code 505 allows only host initiators to connect to the controller port which is a dedicated data replication port.

The PPRC manager module 515 uses containers and services that the VA layer 520 exports. PPRC manager 515 uses interfaces between host port initiator module 510 and VA module 520 for signaling, transfer initiation, and transfer completions. PPRC manager 515 is responsible for managing functions including initiating the connection and heartbeat with the remote controller and initiating the remote copy for incoming host writes (via host port initiator 510); initiating I/O operations for performing full copy, log, and merge; handling error recovery (link failover) and peer communication; and maintaining state information. Device Services layer 525 handles the physical I/O to external devices including the local data storage array and switch.

Inter-Site Controller Heartbeat Timer Operation

Figure 6A is an exemplary flow diagram showing the operation of two of the array controller 'heartbeat' timers. The operation described in Figure 6A is best understood in conjunction with reference to the system architecture shown in Figures 2 and 3. In the embodiment described in Figure 6A, during the course of normal system operation, host computer 101 sends requests to write data to array 203 via controller A1 (201). At step **600**, in response to a write request, array controller A1 sends a write command and the host write data to target array controller B1 via fabric 103A (referred to as 'link 1" in Figure 6), so that the data is backed up on array 213. At step **605**, controller A1 starts a command ('heartbeat') timer which keeps track of the time between issuance of the write command and a response from the target controller B1. If link 1 and controller B1 are operational, then controller B1 writes the data to array 213 and, at step **610**, sends an acknowledgement ('ACK') back to controller A1 via link 1, indicating successful completion of the command.

Asynchronously with respect to the command timer described above, at step **601**, controller A1 may also periodically send a Fibre Channel 'echo' extended link service command to controller B1 via link 1. In one embodiment of the present system, the link echo is sent every 10 seconds; however, the exact frequency of the echoes is not critical, nor is it necessary to have the echoes synchronized with any specific source. At step **603**, controller A1 sets a second 'heartbeat' timer or counter, which can simply be a counter which counts-down using a clock to keep track of the time elapsed since the sending of the link echo. At step **610**, in the normal course of operation, controller A1 receives an 'ACK' from controller B1, indicating that link 1 is operational. The command and link timers are preferably set to time out at intervals which are best suited for the cross-link response time between controllers A1 and B1. It is to be noted that a single inter-site link/command timer may be employed in lieu of the two timers described above. A periodic 'echo' and associated timer may entirely supplant the command timer, or, alternatively, the echo timer may be replaced by the use of a single timer to ensure that each command sent over each inter-site link is responded to within a predetermined time.

At step **615**, due to a failure of link 1 or controller B1, at least one of two situations has occurred - (1) controller A1's command timer has timed out, or (2) controller A1's link timer has timed out. In either event, a link failover operation is initiated. At step **620**, controller A1 transfers control to controller A2, causing A2 to assume control of backup activities. Next, at step **625**, controller A2 proceeds to back up data on storage array 213 by communicating with controller B2 via link 2 (fabric 103B). Since controller B2 shares storage array 213 with controller B1, at step **630**, B2 now has access to the volume (e.g., LUN X') which was previously created by controller B1 with data sent from controller A1. The failover process is further described below with respect to Figure 6B.

Intra-Site Controller Heartbeat Timer Operation

Figure 6B is a flow diagram showing the operation of controller-based 'heartbeat' timers, wherein a controller failover operation is effected by a 'surviving' controller. In the example illustrated in Figure 6B, controllers A1 (201) and A2 (202) are interchangeably represented by the letters 'C' and 'C!', where "C!" represents C's 'companion' controller, i.e., where controller C can be either controller A1 or A2, and controller C! is the companion controller A2 or A1, respectively. This terminology is chosen to illustrate the symmetrical relationship between the two controllers. In the present example, the data from host computer 101 is sent over C's link (e.g., link 1) to a backup volume (e.g., LUN X) via its counterpart controller (e.g., controller B1) at the remote target site.

Initially, at step **635**, controllers C and C! set a 'controller heartbeat' timer or counter to keep track of the time elapsed between receiving consecutive heartbeat signals (hereinafter referred to as 'pings') from the other controller. The controller heartbeat timer is set to time out at a predetermined interval, which allows for a worst-case elapsed time between receiving two consecutive pings from the other controller. Next, during normal operation, at step **640**, controllers C and C! periodically send pings to each other via DUARTs (Dual Asynchronous Receiver/Transmitters) located at both ends of bus 330. Assuming that neither controller C nor controller C!'s heartbeat timer has timed out, at step **643**, both controllers C and C! receive a ping from their companion controller. Both controllers then reset their heartbeat timers at step **645**, and each controller awaits another ping from its companion controller.

In the situation where, for example, controller C fails (step **647**), allowing controller C!'s heartbeat timer to time out (at step **650**), then, at step **655**, controller C! initiates a controller failover operation to move the target LUN on remote storage array to the other controller (e.g., from controller B1 to controller B2). At step **660**, controller C! proceeds by sending backup data to alternate controller (e.g., controller B2) via the alternate link (e.g., link 2). At this point, controller C! has access to the backup volume (e.g., LUN X′) on array 213.

Intra-Site Controller Heartbeat Timer Operation

Figure 6B is a flow diagram showing the operation of a third system (controller-based) 'heartbeat' timer wherein a failover operation is effected in response to a controller failure. As shown in Figure 6B, initially, at step **645**, controller A2 sets a 'heartbeat' timer or counter to keep track of the time elapsed between heartbeat signals from controller A1. Again, in an exemplary embodiment, the length of the heartbeat timer is set to 2 'heartbeats', or 20 seconds; however, the timer can be set to time out in a shorter interval. Next, during normal operation, at step **650**, controller A1 (201) periodically sends 'heartbeat' signals to controller B1 (202) via DUARTs (Dual Asynchronous Receiver/Transmitters) located at both ends of bus 330. At step **655**, assuming that controller A2's heartbeat timer has not timed out, A2 resets its heartbeat timer, and awaits another heartbeat signal from controller A1.

In the situation where, for example, controller A1 fails, allowing controller A2's heartbeat timer to time out, at step **660**, in the absence of receiving a heartbeat signal from controller A1, then at step **665**, controller A2 initiates a link failover operation. At step **670**, controller A2 proceeds by sending backup data to controller B2 via link 2 (fabric 103B). At this point, B2 has access to the backup volume (e.g., LUN X′) on array 213.

Connection Setup

When a remote copy set is bound, connection setup is initiated. In a switched Fibre Channel environment, an initiator controller's host port initiator module 510 (Figure 5) performs discovery to 'find' the target controller. The host port module 510 must use the Fabric's FC-NameServer in order to find controllers which are part of the present system 100. Initially, the user specifies a "target name" which uniquely identifies the remote controller and unit. Once the connection has been setup, a full copy from the initiator unit to the target unit is initiated. The target's data is protected from host access, by the user pre-setting access IDs.

Steady State Operation

Steady state operation of the present system 100 is possible in two modes, synchronous or asynchronous. When the present system is in synchronous mode, the remote data is consistent with the local data. All commands that are returned to the host as completed, are completed on both the initiator and the target. When system 100 is in asynchronous mode, the remote site may lag behind by a bounded number of write I/O operations. All commands that are returned to the host as completed, are completed on the initiator, and may or may not be completed on the target. From a recovery viewpoint the only difference between the operation modes is the level of currency of target members.

Synchronous System Operation

Figure 7 is a flowchart showing synchronous system operation. In synchronous operation mode, data is written simultaneously to local controller cache memory (or directly to local media if the write request is a write-through command), as well as to the remote subsystems, in real time, before the application I/O is completed, thus ensuring the highest possible data consistency. Synchronous replication is appropriate when this exact consistency is critical to an application such as a banking transaction. A drawback to synchronous operation is that long distances between sites mean longer response times, due to the transit time, which might reach unacceptable latency levels, although this situation is somewhat mitigated by write-back cache at the target. Asynchronous operation, described in the following section, may improve the response time for long-distance backup situations.

Steady state synchronous operation of system 100 proceeds with the following sequence. As shown in Figure 7, at step **701**, host computer 101 issues a write command to local controller A1 (201), which receives the command at host port 109 over path 221h at step **705**. At step **710**, the controller passes the write command down to the VA level software 530 (Figure 5) as a normal write. At step **715**, VA 530 writes the data into its write-back cache through the normal cache manager path (i.e., through the device services layer 525). On write completion, VA 530 retains the cache lock and calls the PPRC manager 515. At step **720**, PPRC manager 515 sends the write data to remote target controller B1 (211) via host port initiator module 510. The data is sent through the remote copy dedicated host port 109 via path 221D, and across fabric 103A. Next, at step **725**, remote target controller B1 writes data to its write-back cache (or directly to media if a write through operation). Then, at step **730**, controller B1 sends the completion status back to initiator controller A1. Once PPRC manager 515 in controller A1 has received a completion status from target controller, it notifies VA 530 of the completion, at step **735**. At step **740**, VA 530 completes the write in the normal path (media write if write through), releases the cache lock, and completes the present operation at step **745** by sending a completion status to the host 101. The cache lock is released by the last entity to use the data. In the case of a remote write, the cache is released by the PPRC manager upon write completion.

Asynchronous System Operation

Figure 8A is a flowchart showing asynchronous operation the present system 100. Asynchronous operation provides command completion to the host after the data is safe on the initiating controller, and prior to completion of the target command. During system operation, incoming host write requests may exceed the rate at which remote copies to the target can be performed. Copies therefore can be temporarily out of synchronization, but over time that data will converge to the same at all sites. Asynchronous operation is useful when transferring large amounts of data, such as during data center migrations or consolidations.

Asynchronous operation of the present system 100 proceeds with the following sequence. As shown in Figure 8A, at step **801**, host computer 101 issues a write command to local controller A1 (201), which receives the command at host port 109 over path 221h at step **805**. At step **810**, the controller passes the write command down to the VA level software 530 (Figure 5) as a normal write. At step **815**, VA 530 writes the data into its write-back cache through the normal cache manager path **(i.e., through the device services layer 525)**. On write completion, VA 530 retains the cache lock and calls the PPRC manager 515. At step **820**, PPRC Manager "micro-logs" the write transfer LBN extent, as well as the command sequence number and additional context in the controller's non-volatile write-back cache 'micro-log'. This is done in all situations (not just in error situations), in case the initiator controller (A1) crashes after status is returned to the host, but before the remote copy completes. A small reserved area of cache is dedicated for the micro-log.

Micro-logging is done during steady state operation for each asynchronous transfer, not just during error situations. The micro-log information is only used when the controller crashes with outstanding remote copies (or with outstanding logging unit writes). The micro-log contains information to re-issue ('micro-merge') the remote copies by either the 'other' controller (in this example, controller A2) upon controller failover, or when 'this' controller (A1) reboots, in the situation wherein both controllers A1 and A2 are down.

At step **825**, PPRC manager 515 calls back VA 530 to complete the host write request, and the host is given the completion status. VA 530 retains the cache lock and Data Descriptor data structure. At step **830**, PPRC manager 515 (via host port initiator module 510) sends the write data to the remote target. Order preserving context is also passed to host port initiator module 510. At step **835**, remote target controller B1 (211) writes data to its write-back cache (or associated media if a write-through operation). A check is then made by controller A1 at step **840** to determine whether the remote copy successfully completed. If so, then, at step **845**, target controller B1 sends the completion status back to initiator controller A1. At step **850**, PPRC manager 515 marks the micro-log entry that the write has completed. The PPRC manager also unlocks the controller cache and de-allocates the Data Descriptor.

If, at step 840, if it was determined that the remote copy operation did not complete successfully, then at step **855**, if the initiator controller (A1) failed while the remote copy was in transit, then a 'micro-merge' operation (described below with respect to Figure 8B) is performed. If the remote copy was unsuccessful for other reasons, then at step **860**, other error recovery procedures (not part of the present disclosure) are invoked.

Figure 8B is a flowchart showing a 'micro-merge' operation. A micro-merge operation is applicable during asynchronous operation when the controller has failed in the window where the host write status has already been returned to the host, but where the remote copy operation (or write history log operation) has not completed. As indicated above, these 'outstanding' writes were logged to the initiator controller A1's write-back cache which is also mirrored in partner controller A2's (mirrored) write-back cache, so that the cache data is available to controller A2 if controller A1 fails. If a controller failover has taken place (as explained in the next section, below), then the partner controller (A2) re-issues these remote copies from the micro-log. Alternatively, if both controllers A1 and A2 are down, then controller A1 itself re-issues these writes when it restarts.

The following sequence takes place in the controller during micro-merging mode. At step **865**, access to the initiator unit by the host is inhibited until the micro-merge is complete. At step **870**, for each valid entry in the micro-log in the controller write-back cache, the initiator unit is read at the LBN described. If the read has an FE (Forced Error), then the FE will be copied to the target (which is highly unlikely, since the area was just written). If the read is unrecoverable, then the target member is removed, because it is impossible to make the target consistent. If the read is successful, the data is then written to the remote target member using the normal remote copy path. Alternatively, if write history logging is active, the data is written to a log unit, as described below in the 'Write History Logging' section.

In addition to command and LBN extent information, the micro-log contains the command sequence number and additional context to issue the commands in the same order received from the host. At step **875**, if the remote copy of the entry was successful, then at step **880**, the recorded entry in the micro-log is cleared, and the next entry is 're-played', at step 870. If the remote copy of the entry was not successful, then at step **895**, then error recovery procedures (not part of the present disclosure) are invoked. After completing all micro-merges (step **885**), the initiator unit is made accessible to the host at step **890**.

Link Failover

'Link failover' is recovery at the initiator site when one of the two links has failed. Examples of a link failover situation include a target controller rebooting, a switch failure, or an inter-site link failure. In a first situation, if the initiator controller has two consecutive failed heartbeats and its dual partner has two consecutive successful 'heartbeats', then a link failover is performed. It may also performed in a second situation wherein a remote write has failed due to a link error and its dual partner last had two successful heartbeats (a failed write is held for two successive heartbeats).

Figure 9 is a diagram showing an example of a link failover operation. As shown in Figure 9, link 901 is lost to initiator controller A1. In the present example, controller A1 is in communication with partner controller A2, which indicates to A1 that A2's link 902 to controller B2 is operational. In this situation, initiator controller A1 attempts link failover recovery procedures by attempting to communicate through its dual redundant partner controller A2 and resume operations. In one embodiment of the present system, a link failover is accomplished by restarting (re-booting) controller A1, to force the initiator unit X on array 203 from controller A1 to its partner controller A2. Once unit X is moved over from controller A1 to controller A2 on the initiator side, controller A2 then 'pulls' target unit Y over to its dual redundant partner B2 where controller A2 (the 'new' initiator) can access it. Link failover is not performed upon receiving SCSI errors (unit failures) from the remote unit, because the other controller will likely encounter the same error. It is to be noted that the initiator controllers (A1 and A2) control the entire failover operation (the target controller, e.g., B2 is the slave).

Operations resume between controllers A2 and B2 if the previous steps were successful. When link failover is successful, the host retries any writes, similar to a controller failover event. Incoming writes during this time are not queued, but rather rejected, so the host will retry them. If the link is restored, the host can move the unit back to the original side.

Write History Logging

The present system 100 provides a unique storage set (typically, RAID level 1, level 0+1, or level 5 storage set) that is considered as a logical unit by the associated controller, and which is employed to create a write history (transaction) log comprising log commands and extents, as well as data, during situations where the remote member of a remote copy set ('RCS') is not available. This storage set, called a 'log unit', hereinafter, is subsequently 'replayed', in the exact same order in which it was written, to the remote RCS member to merge the local and remote RCS members. The log unit is preferably located on the same storage array as the local remote copy set member.

Figure 10 is a high-level flow diagram showing a write history log operation performed by the present system 100 when both links are down, or when the remote site is down. Thc top section of Figure 10 depicts normal operation of the present system 100, where arrow **1005** shows write data from host computer 101 being stored on local (initiator) array 203. Arrow **1010** indicates that the write data is normally backed up on remote (target) array 213. The lower section of Figure 10 shows system 100 operation when the links between the local and remote sites are down, or when the remote pair of array controllers 211/212 are inoperative, and thus array 213 is inaccessible to local site 218, as indicated by the broken arrow **1015**. In this situation, as indicated by arrows **1020**, write operations from the local host (ref. no. 101, shown in Figures 2 and 3), are directed by the initiator array controller (either 201 or 202 in Figures 2 and 3) to both array 203 and log unit 1000.

Extents and data are both written to log unit 1000, the format for which is described in detail below with respect to Figure 13. The logging is done via write through to media. The log unit 1000 is required to have write-back disabled. Enabling write-back would require a DMA copy of the data so that it could be written to media at a later time. The DMA copy process incurs extra overhead, consumes resources, and adds complexity, so write-back mode is not desirable for the present logging function.

A log unit is 'replayed' to the remote site 'partner' controller when the link is restored, the remote site has been restored, or when the local site has been restored (during a site failback, described below with respect to Figure 12). Replaying the log means sending all commands and data to the remote partner in order (for each remote copy set) for all remote copy sets associated with the log unit. A merging operation (hereinafter referred to as simply 'merge') is performed by system 100 to quickly return a remote copy set (both local and remote members) to the same data state (i.e., up to date) after link restoration or remote site restoration. A 'mergeback' operation is performed by system 100 to restore the local site back to the same state as the remote site during site failback, assuming that the initiator site is intact. Log units 1000 and 1001 are used to replay the transactions for the merge and mergeback functions, respectively. Because the present system may avoid having to perform a full copy, particularly when a site is down only temporarily, site failover can be performed for short down-time situations, since site failback/resynchronization is quickly accomplished.

Figure 11 is a flowchart showing an exemplary write history log operation followed by an exemplary merge performed by the present system 100. As shown in Figure 11, at step **1105**, access from site 218 to target array 213 is broken, as indicated by arrow 1015 in Figure 10. At step **1110**, the write history logging operation of the present system is initiated by array controller 201 in response to a link failover situation, as explained above with respect to Figure 9. Initiation of the logging function requires that assignment of a dedicated log unit 1000/1001 has been made by a system user. At step **1115**, write operations requested by host computer 101/102 are redirected by associated initiator array controller 201 (optionally, controller 202) from target controller 211 to log unit 1000. The log unit descriptors reside at the beginning of the unit. The extent entries are logged before the data in a spiral fashion. Figure 13, described below, shows the format of data and extent information stored on a log unit.

The present system allows different logging streams active at the same time to be intermixed. The log unit is not partitioned in any manner based on the presence of different log streams. If asynchronous operation is enabled, then asynchronous writes occur to the log unit, wherein completion status is returned prior to writing the log unit.

A step **1120**, access to target array 213 is re-established, and at step **1125**, the merging operation is initiated. At step **1130**, the data and extent information from host computer 101 is still written to log unit 1000, but the host writes are delayed to allow the merge to catch up to the log writes. More specifically, the controller turns on a 'command throttle' to slow host I/O down, so the merge can make progress. Then at step **1135**, a data read stream is started at the appropriate LBN of the log unit. The data is written to the remote target member using the normal remote copy path. The command order is preserved with the context stored in the log unit. A description of the method used by the present system to preserve command order is presented with respect to Figure 14, below. At step **1140**, writes to the log unit 1000 are paused, allowing merge read operations to completely catch up. At this point, there must be more merging I/O operations performed than host I/O operations to avoid reaching the end of the log unit. Therefore, when the merge stream catches up to the log stream, host writes are quiesced (temporarily queued) to make the transition. At step **1145**, the merge reads catch up with the log writes. Finally, at step **1150**, the log and merge operations are completed, and at step **1155**, normal backup operation of system 100 is resumed.

Note that during a merge operation, it is not sufficient to send the data over the wire in an order compatible with the original write ordering - the data has to be written to "media" (either magnetic media or controller write-back cache) at the remote site in compatible order. This means the local and remote controllers have to control the number of outstanding write operations so that the execution of these commands cannot be reordered, even in the presence of Fibre Channel errors in the inter-controller link, to pervert this order.

The present system merges the write commands in the proper order, including write commands which are allowed to overlap each other. For example, if during logging, the original host writes command A and it completes before it issues command C, then during merging, the "play back" must also finish command A before starting command C.

Figure 12 is a flow diagram showing a log operation in response to a site failover. As shown in Figure 12, during the course of normal operations, host writes to array 203 are backed up on the corresponding remote copy set LUN in array 213, as indicated by arrow **1210**. If, for example, array 203 becomes inaccessible by local host 101, as indicated by arrow **1215**, then site failover is performed, since host 101 cannot write the local array 203, and the controllers at both sites cannot communicate with each other, so inter-site backup is not possible, as shown by arrow **1220**. When site failover takes place, a new remote copy set is created with the original target member as the new initiator (now at site 219), and the original initiator as the new target member. The remote set consists of two members, as before the site failover. The new initiator unit now presents the WWID of the original initiator (remote copy set's WWID) to the host at site 219. In this situation, write operations from the host (102) at site 219 are directed by the initiator array controller (either 211 or 212, Figures 2 and 3) to array 213, as indicated by arrow **1225**, and to log unit 1001, as shown by arrow **1230**.

Upon site failback, merge-back takes place. The merge-back operation is analogous to the merge operation described with respect to Figure 11, except that the 'initiator' unit during the merge-back operation is the LUN resident on array 1001, and the 'target' is the original initiator LUN. Initiator control is then moved back to the original initiator site after communication between sites.

Figure 13 is a diagram showing an exemplary format 1300 of data and extent information stored on a log unit 1000/1001. As shown in Figure 13, the Log Container (log unit) Descriptor ('LCD') 1301 starts at logical block address (LBA) 0, and is used to describe attributes of the log 'container' (i.e., the log unit). The LCD 1301 contains information comprising the log disk state (free, allocated, or in use), the current log position, and membership information, including (1) the initiator LUN ID, (2) the target LUN ID, and (3) the target name. The Log Stream Descriptor ('LSD') 1302 is used to describe the merging state and includes information such as the current log stream state (free, normal, logging, or merging), the current merge position, and target information which is bit-encoded to denote their specific LCD membership parameters. Following the LCD 1302 is a series of Extent Descriptor List/data segment pairs 1303*/1305*, starting at LBA 100. The Extent Descriptor List ('EDL') 1303* (where '*' denotes the rank in the series) is used to describe the host write data, and includes an Extent Descriptor header and Extent Descriptor array [*] member information. The Extent Descriptor header contains pointers 1306 to the next EDL and the previous EDL, more specifically, the next/previous EDL logical block address (LBA). The Extent Descriptor array member information includes (1) the LBA of the data at the target destination; (2) a pointer 1307 to the associated data segment 1303 on the log unit (the data LBA); (3) bit-encoded LCD membership parameters for the target(s), (4) a 'look-ahead limit' used to describe the host data write ordering, and (4), the block count for the associated data in the data segment 1303 following the particular EDL. The terminal EDL segment is indicated by a 'next EDL' LBA (flag) value of -1.

Command Ordering

The present system 100 employs a method to ensure the proper ordering on the remote media. This method utilizes the inventive concept that the order of writes to the media at the remote site don't have to exactly match their original order of occurrence. Rather, it is sufficient merely to ensure that if write A completed before write B started at the primary site, that write B cannot be recorded to media before write A at the remote site.

Figure 14 is a flowchart illustrating an exemplary method used by the present system to ensure proper write ordering on the remote media. As shown in Figure 14, at step **1405**, host computer 101 issues a write command to initiator controller A1 (Figure 3). At step **1410**, each write command is assigned a sequence number when it is received by the controller; i.e., the Command Identifier ('CMD ID') for the present command is set to the current Sequence Number ('SQN') The initial sequence number has a value of 1, and subsequent sequence numbers are generated by incrementing a 16 bit sequence number counter. Accordingly, the current SQN is incremented at this point. At step **1412**, controller A1 receives write data from host computer 101. Next, at step **1415,** when a command (i.e., the write to the local storage array) completes, the current value of the sequence number SQN is also stored in the command's control block. This value is called the 'look-ahead limit' ('LAL') for this command. At step **1420**, a write completion status message is sent to host 101.

At step **1430**, an InProgress Queue is checked to determine if any writes are in progress. If not, then at step **1433**, controller A1 sets the value of its presently stored 'look-ahead fence' ('LAF') to the command's look-ahead limit (CMD LAL), and the process proceeds with step **1455**, below. If there are writes presently in progress (at step **1430**), then at step **1440**, if the CMD ID is less than the value of the present look ahead fence, then at step **1445**, the look-ahead fence is set to the minimum of the old look-ahead fence and the command's look-ahead limit. At step **1455**, the write command is put into the InProgress Queue, and the command is sent to remote controller B1, at step **1457**. From this point,
the process continues at step **1460**, below.

If, at step **1440**, if the CMD ID is not less than the value of the present look ahead fence, then at step **1442**, the write command is placed in a queue of pending commands (the 'Pending Queue'). Next, at step **1460**, controller A1 waits for remote controller B1 to complete a write operation. At step **1462**, whenever a write operation completes on the remote controller, the corresponding write command is removed from the InProgress Queue, at step **1463**. Next, at step **1465**, if the completed write command's look-ahead limit is not equal to the current look-ahead fence, then the process loops back to step **1460**, where controller A1 waits for remote controller B1 to complete a write operation. If, at step **1465**, the completed write command's look-ahead limit is equal to the current look-ahead fence, then, at step **1467**, if the InProgress Queue is empty, processing continues at step **1475**, below. If, at step **1467**, if the InProgress Queue is not empty, then controller A1 replaces the look-ahead fence with the minimum value of the look-ahead limits of all currently outstanding writes (i.e., the minimum value of the CMD LAL in the InProgress Queue) at step **1470**.

At step **1475**, command processing continues. The remote controller feeds all the commands in the waiting queue back into the previous step **1430** of the present algorithm as if they were new commands to start execution of any waiting commands that pass the new look-ahead fence criteria. In a preferred embodiment, signed 16-bit arithmetic is used to perform comparisons and determine minimums; therefore, the present method works properly even when the counters wrap around, as long as there are less than 32768 outstanding I/O operations.

Merges are done in a manner similar to the process described above. After the write command completes on the primary controller (controller A1, in the present example), the primary controller notices that the link is marked as down; the controller then puts the write command (and its data) into the log, including the command sequence number and look-ahead limit. When the primary controller 'replays' the log over the link to remote controller B1 after the link comes up again, the remote controller executes the same algorithm as above to determine when it can execute the write commands. Merge writes (step **1425**) are processed, beginning at step **1430,** by the algorithm described above (with respect to Figure 14). This algorithm allows the maximum parallelism of write commands in the remote controller without risking write order inversions, and without requiring changes in the Value Added (VA) code to detect errors during data fetches.

The present system 100, as illustrated by the following example, merges the write commands in the proper order, including write commands which are allowed to overlap each other. For example, if during logging, the original host writes command A and it completes before it issues command C, then during merging, the "play back" must also finish command A before starting command C.

In an attempt to have as many outstanding commands to the merging member as possible, the relationship between the start and end of write commands is recorded during logging. This relationship takes the form of a first command sequence number that is incremented and stored for each new write command. Then, as the write command completes, the next command sequence number is noted and stored. This stored value is the 'look-ahead limit'. The difference between these two values (the command sequence numbers) is the number of new write commands issued while the original write command was in process.

In the following discussion of Tables 1 and 2, "LAL" = 'look-ahead limit, "WHL" = 'write history log, and "CMD SQN" = 'command sequence number'. For the present example, assume that during logging, the host writes shown in Table 1 take place.

TABLE 1

WriteA starts	CMD SQN= 1
WriteB starts	CMD SQN= 2
WriteA finishes	(WHL entry of WriteA - CMD SQN1, LAL = 2)
WriteC starts	CMD SQN= 3
WriteB finishes	(WHL entry of WriteB - CMD SQN2, LAL = 3)
WriteD starts	CMD SQN= 4
WriteD finishes	(WHL entry of WriteD - CMD SQN4, LAL = 4)
WriteE starts	CMD SQN= 5
WriteC finishes	(WHL entry of WriteC - CMD SQN3, LAL = 5)
WriteE finishes	(WHL entry of WriteE - CMD SQN5, LAL = 5)

Next, assume that during merging, the sequence of writes shown in Table 2 occur:

TABLE 2

1. WriteA, CMD SQN1, LAL = 2 starts
2. WriteB, CMD SQN2, LAL = 3 starts
3. WriteB finishes.
4. WriteA finishes.
5. WriteC, CMD SQN3, LAL = 5 starts
6. WriteD, CMD SQN4, LAL = 4 starts
7. WriteD finishes
8. WriteE, CMD SQN5, LAL = 5 starts
9. WriteC finishes
10. WriteE finishes

Initially, the Current LAL is uninitialized. The following section describes the look-ahead limit (LAL) for each step [n] shown in Table 2.

Step	Look-Ahead Limit
[1]	Current LAL = 2.
[2]	Current LAL = 2 (lowest LAL of the outstanding commands). At this point, WriteC and subsequent writes can not start, since their CMD IDs are greater than the Current LAL.
[3]	Current LAL remains at 2 since it is the lowest LAL of the outstanding commands.
[4]	Current LAL goes to uninitialized as there are no more outstanding commands.
[5]	Current LAL = 5.
[6]	Current LAL = 4 (lowest LAL of the outstanding commands). WriteE and subsequent writes cannot start since their CMD IDs are greater than the Current LAL.
[7]	Current LAL = 5.
[8]	Current LAL = 5.

[9] Current LAL = 5.

[10] Current LAL = uninitialized.

Figure 15 is a flowchart of a method used by the present system to calculate a worst-case response time for effecting failover operations. In the operation of this method, a set of test frames are sent between local and remote sites, for example, between sites 218 and 219 in Figure 2. When these frames are sent, the time in which it takes for a response determines the time delay over the link. This response time is related to distance between the sites. Initially, as shown in Figure 15, at step **1500**, a point to point connection is established between each site, for example, between initiator (local) controller A1 at site 218 and target (remote) controller B1 at site 219. Given that each port for DRM operations on the arrays is dedicated, fabric congestion is not a particular concern for the test frame timing.

At step **1505**, 2 or more test frames are sent from controller A1 to controller B1. Upon receipt of each test frame by controller B1, then at step **1510**, a response is then sent from controller B1 back to controller A1. At step **1515**, the average transit time is then calculated for the two-way transmit/response path between controllers A1 - B1 - A1. This average two-way transit time is then multiplied by a contingency factor (at step **1520**) to produce the timeout used by the controller (at step **1525**) for worst case timing between the links, given busy time in the controller as well as potential transient error effects. In an exemplary embodiment of the present system 100, the contingency factor is preferably 100, but other values may be set by a system user in accordance with such factors as the probability and duration of unusual propagation delays and the like.

Although values for the worst-case transit time and the look-ahead limits are automatically established by the system controller(s), the transit timeout and 'number of outstanding command' values are user-adjustable, which allows further tuning of the inter-site links as a function of system performance requirements, and to compensate for factors other than distance.

Although the above description refers to specific embodiments of the invention, the invention is not necessarily limited to the particular embodiments described herein. It is to be understood that various other adaptations and modifications may be made within the spirit and scope of the invention as set forth in the appended claims.